Mathematical Programming 64 (1994) 17–51

Dynamic factorization in large-scale optimization

Gerald G. Brown*, Michael P. Olson

Naval Postgraduate School, Monterey, CA 93943, USA

(Received 20 July 1990; Revised manuscript received 23 March 1993)

Abstract

Factorization of linear programming (LP) models enables a large portion of the LP tableau to be represented implicitly and generated from the remaining explicit part. Dynamic factorization admits algebraic elements which change in dimension during the course of solution. A unifying mathematical framework for dynamic row factorization is presented with three algorithms which derive from different LP model row structures: generalized upper bound rows, pure network rows, and generalized network rows. Each of these structures is a generalization of its predecessors, and each corresponding algorithm exhibits just enough additional richness to accommodate the structure at hand within the unified framework. Implementation and computational results are presented for a variety of real-world models. These results suggest that each of these algorithms is superior to the traditional, non-factorized approach, with the degree of improvement depending upon the size and quality of the row factorization identified.

Key words: Factorization; Linear programming; Generalized upper bounds; Pure networks; Generalized networks

1. Introduction

A recurring theme in the development of algorithms for linear programming has been the identification and exploitation of special problem structure. Ideas as apparently disparate as the bounded-variable simplex method, primal and dual decomposition methods, pure and generalized network primal simplex algorithms, primal partitioning and column generation schemes may be unified to a degree with this view.

The factorization approach introduced by Graves and McBride (1976) isolates special structure in LP tableaus. We are interested in using factorization to reinterpret existing algorithms, and to discover common principles and apply them to develop new algorithms.

*Corresponding author.

SSDI 0025-5610(93)E0044-F

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1994		**00-00-1994 to 00-00-1994**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Dynamic factorization in large-scale optimization	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Operations Research Department,Monterey,CA,93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Factorization of linear programming (LP) models enables a large portion of the LP tableau to be represented implicitly and generatedfrom the remainingexplicit part. Dynamicfactorization admits algebraicelementswhichchangein dimensionduring the courseof solution.Aunifyingmathematical framework for dynamic row factorization is presented with three algorithms which derive from differentLPmodelrowstructures: generalizedupperboundrows, purenetworkrows,and generalized networkTOWS. Eachof these structuresis a generalization of its predecessors, and each corresponding algorithm exhibitsjust enough additional richness to accommodate the structure at hand within the unifledframework. Implementation andcomputational results arepresentedfor a varietyof real-world models. Theseresultssuggestthateachof thesealgorithmsis superiorto the traditional, non-factorized approach, with thedegreeof improvement dependinguponthesizeandqualityof therowfactorization identified.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**35**	

Although all algorithms developed this way will, in theory, solve any LP, the efficiency of any particular factorization approach will be influenced by the relative number of factored constraints and their influence on the algorithm: the *size* and *quality* of the special structure isolated determines the influence of any particular factorization applied to any particular LP.

Based on prior work by Brown and Graves (1975), in which generalized upper bound rows were successfully incorporated in a large-scale optimization system, we are interested in pursuing *dynamic* row factorization, where the dimension of the factored structure may vary (or even fail to be present) as the solution progresses. In our setting, we require the row structure of the model instance to be specified prior to solution, and that this structure remain fixed during solution. An extension of this approach is to allow the row structure to vary as the model is solved: this is a conceptually simple extension of the approach.

Each algorithm is developed by factoring the constraints of the LP model into two classes: those that have the special structure (*factored*) and those that do not (*explicit*). This constraint factorization induces a factored structure in the LP tableaus which is exploited computationally. We demonstrate the dynamic factorization approach for three special structures:

- generalized upper bound rows;
- pure network rows; and
- generalized network rows.

We implement each of the factorization algorithms by integrating it within the X-System (Brown and Graves, 1975).

While the terms "partitioning" and "factorization" are frequently used interchangeably in the literature, we observe a distinction between the two approaches. We consider *partitioning* methods to be based on special structure in the original problem instance, which need not induce special structure in the LP tableau — in fact, the method need not be tableau-based. In contrast, *factorization* methods are based on special structure which occurs in bases and thus in the basic tableau. Thus, we classify dual decomposition (Dantzig and Wolfe, 1960), primal decomposition (Benders, 1962), and primal partitioning (Rosen, 1964) as examples of partitioning methods.

Perhaps the earliest example of what we consider factorization is the treatment of simple upper bounds by Dantzig (1954) and (1963) and, independently, by Charnes and Lemke (1954). They observe that it is more efficient to enforce the "logical" upper bound constraints with logical tests within the algorithm rather than treat them explicitly along with other "structural" constraints. While not originally presented in the context of a formal tableau factorization, the approach is easily viewed as such.

The mutual primal–dual method of Graves (1965) focuses attention on the special role of nonnegativity constraints in linear programming. A clear distinction is drawn between the computational convenience of treating nonnegativity constraints implicitly rather than

explicitly and the unambiguous mathematical equivalence of all problem constraints, structural or nonnegativity. Emphasizing the special importance of inequality constraints, the approach yields an elegant theory and, as we will see, efficient implementations. We view this algorithm as the first formal example of factorization.

A similar primal–dual algorithm is presented by Balinski and Gomory (1965). Related work, in which efforts are made to exclude slacks from the product-form representation of the primal basis, includes that of Zoutendijk (1970) and Powell (1975).

Dantzig and Van Slyke (1967) extend the earlier work for simple upper bounds and lend a more structured treatment to generalized upper bounds (GUB). In a problem with p GUB constraints and m structural constraints, their approach requires a working basis of dimension $m+1$, a considerable savings when p is large.

Hartman and Lasdon (1972) specialize this approach to the multicommodity capacitated transshipment problem. In this case, the structure of the basic pure network columns introduces additional structure into the working basis, allowing further simplifications in basis representation and update techniques. Helgason and Kennington (1977) develop techniques for representing the working basis in product form and provide graphic interpretation of the basis updates. Kennington (1977) reports an implementation of the algorithm.

McBride (1972) and Graves and McBride (1976) formalize and generalize the factorization approach. They view factorization as a unifying framework for tableau-based simplex specializations and illustrate this by developing a variation of the GUB algorithm of Dantzig and Van Slyke and a GUB algorithm for the doubly-coupled linear programs of Hartman and Lasdon (1970). They present a new algorithm for the set partitioning LP and an equality-constrained form of the pure network with side constraints model. Brown and Graves (1975) report an implementation of inequality-form, dynamic GUB row factorization for large-scale problems.

Schrage (1975) extends the succession of simple and generalized upper bounds by introducing variable upper bounds (VUB), which are constraints of the form $x_j \leqslant x_k$, where x_k is said to be the variable upper bound of x_j. Schrage implicitly represents the VUB constraints by expressing VUB variables in terms of other variables. This permits the basis representation to be treated in two parts, one a large matrix which changes infrequently and thus needs only occasional update, and the other a small working basis which requires regular attention. Thus, computation and storage savings may be realized. Schrage (1978) extends these ideas to what he calls generalized VUB (GVUB) constraints, which arise frequently in models with fixed charges.

Klingman and Russell (1975) sketch a factorization method for solving transportation problems with side constraints. They suggest techniques for performing simplex iterations and updating the problem representation. Chen and Saigal (1977) present a similar approach for solving capacitated network flow problems with additional linear constraints. Both of these presentations employ a graphical description of the basis update and treat the basis in two parts: one corresponding to a rooted spanning tree defined on the underlying graph, and the other a general working basis. Glover et al. (1978) report an implementation of the

Klingman and Russell design, but one which (curiously) only accommodates a single side constraint. McBride (1989) reports an implementation which requires the pure network rows to be equalities and allows more than one side constraint.

Generalized networks with side constraints are addressed by Hultz and Klingman (1976), who present details for the simplex priceout, column generation, and basis update. Hultz and Klingman (1978) report an implementation that (curiously) solves the "singularly constrained" generalized network problem. McBride (1989) reports an implementation that is not restricted to a single side constraint.

The factorization approach has been extended by consideration of embedded structures. Glover and Klingman (1981) consider an LP with embedded pure network structure, i.e., the pure network structure appears in only a subset of the rows and columns of the technological coefficient matrix. They give an algorithm similar in spirit to their pure network with side constraint model, but the presence of the "side variables" significantly complicates the basis representation and update. They report an implementation of the algorithm but (curiously) restrict test problems to have no complicating variables.

McBride (1985) solves an LP with embedded generalized network structure, presenting methods for pricing, column generation, basis representation update and data structures. A successful implementation is reported to be about five times faster than MINOS (ca. 1977: Murtagh and Saunders, 1977) for the models tested.

Algorithms to solve problems with special substructures have motivated research to efficiently identify such substructures. Brearley, Mitra and Williams (1975) describe algorithms for detecting GUB row sets and exclusive-row structure sets (a set of rows whose structure may be transformed to GUB by column scaling). Greenberg and Rarick (1974) and Brown and Thomen (1980) develop algorithms to identify GUB sets. Brown and Wright (1984) identify pure network constraint substructures. Brown, McBride and Wood (1985) present a method for locating embedded and row-only generalized network structures.

Todd (1983) develops a geometric interpretation of factorization which is for our purposes equivalent to the algebraic development of Graves and McBride (1976).

In the following sections we establish notational conventions, develop the mathematical foundation of a primal–dual simplex method and show the effects of row-factorization. Next, a general row-factorization algorithm is developed, specialized to GUB, pure network, and generalized network rows, and tested on a suite of real-world problems. Finally, we discuss how the methods can be generalized further and how they can be applied more effectively.

2. Mathematical preliminaries

The traditional statement of the linear programming (LP) problem is

$$\text{(LP)} \quad \min_{y} \; wy$$
$$\text{s.t.} \quad a_i y \leqslant r_i, \quad i = 1, \ldots, m,$$
$$e_j y \geqslant 0, \quad j = 1, \ldots, n,$$

where y is an n-vector of decision variables, w a vector of cost coefficients, each a_i an n-vector of technological transformation coefficients, each r_i a scalar right-hand side coefficient, and e_j the jth unit vector. While this statement of the problem is clear and unambiguous, there are reasons for preferring an alternative. The insistence upon drawing a formal distinction between the "structural" constraints $a_i y \leqslant r_i$ and the "nonnegativity" constraints $e_j y \geqslant 0$ obscures the mathematical structure of the problem by suggesting that the two types of constraints are inherently different. Certainly the exploitation of the special structure of the $e_j y \geqslant 0$ constraints leads to computational efficiencies; however, in our theoretical development of the algorithm, we prefer to treat them simply as general inequality constraints.

In order to achieve a consistent form, we rewrite the nonnegativity constraints as $-e_j y \leqslant 0$ and group them with the structural constraints. The problem statement then becomes

$$\text{(PLP)} \quad \min_{y} \; wy$$
$$\text{s.t.} \quad a_i y \leqslant r_i, \quad i = 1, \ldots, m+n,$$

where wy is called the extremal function.

From the standpoint of a primal algorithm, a matrix partitioned form of the primal tableau is derived. Let $\{a_{i_1}, a_{i_2}, \ldots, a_{i_n}\}$ be a basis for $\mathbb{R}^n$ at y^0. For notational convenience we will partition the constraints into two sets, those that are basic (binding) at y^0 and those that are nonbasic (not necessarily binding) at y^0,

$$B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} a_{i_1} \\ a_{i_2} \\ \vdots \\ a_{i_n} \end{bmatrix}, \qquad f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} r_{i_1} \\ r_{i_2} \\ \vdots \\ r_{i_n} \end{bmatrix},$$

$$D = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix} = \begin{bmatrix} a_{i_n+1} \\ a_{i_n+2} \\ \vdots \\ a_{i_n+m} \end{bmatrix}, \qquad g = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_m \end{bmatrix} = \begin{bmatrix} r_{i_n+1} \\ r_{i_n+2} \\ \vdots \\ r_{i_n+m} \end{bmatrix}.$$

Using this notation, the current basic solution y^0 may be expressed as $By^0 = f$, and since the rows of B are by definition linearly independent, y^0 exists.

To isolate the important algebraic components let us assume that at the current basic solution the basis consists of h structural constraints and $n-h$ nonnegativity constraints. Then

$$B=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_h \\ b_{h+1} \\ b_{h+2} \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} a_{i_1} \\ a_{i_2} \\ \vdots \\ a_{i_h} \\ -e_{j_{h+1}} \\ -e_{j_{h+2}} \\ \vdots \\ -e_{j_n} \end{bmatrix} \quad \text{and} \quad f=\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_h \\ f_{h+1} \\ f_{h+2} \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} r_{i_1} \\ r_{i_2} \\ \vdots \\ r_{i_h} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

In partitioned matrix form,

$$B=\begin{pmatrix} \overset{h}{A_{11}} & \overset{n-h}{A_{12}} \\ 0 & -I \end{pmatrix}\begin{matrix} h \\ n-h \end{matrix},$$

and thus

$$P=-B^{-1}=\begin{pmatrix} \overset{h}{A_{11}^{-1}} & \overset{n-h}{-A_{11}^{-1}A_{12}} \\ 0 & I \end{pmatrix}\begin{matrix} h \\ n-h \end{matrix}.$$

Similarly, D can be written as

$$D=\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_h \\ d_{h+1} \\ d_{h+2} \\ \vdots \\ d_m \end{bmatrix} = \begin{bmatrix} -e_{j_1} \\ -e_{j_2} \\ \vdots \\ -e_{j_h} \\ a_{i_{h+1}} \\ a_{i_{h+2}} \\ \vdots \\ a_{i_{n+m}} \end{bmatrix} \quad \text{and} \quad g=\begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_h \\ g_{h+1} \\ g_{h+2} \\ \vdots \\ g_m \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ r_{i_{h+1}} \\ r_{i_{h+2}} \\ \vdots \\ r_{i_{n+m}} \end{bmatrix},$$

and in partitioned matrix form

$$D=\begin{pmatrix} \overset{h}{-I} & \overset{n-h}{0} \\ A_{21} & A_{22} \end{pmatrix}\begin{matrix} h \\ m-h \end{matrix}.$$

Then

$$DP=-DB^{-1}=\begin{pmatrix} \overset{h}{A_{11}^{-1}} & \overset{n-h}{A_{11}^{-1}A_{12}} \\ -A_{21}A_{11}^{-1} & A_{22}-A_{21}A_{11}^{-1}A_{12} \end{pmatrix}\begin{matrix} h \\ m-h \end{matrix},$$

and we shall call DP the *principal part* of the tableau. By partitioning $w=(w_1, w_2)$, $g^{\mathrm{T}}=(g_1, g_2)^{\mathrm{T}}$ and $y^0=(y_1^0, y_2^0)$, the *complete tableau* may be written in partitioned matrix form as

$$\begin{array}{c} \begin{matrix} h & n-h & 1 \end{matrix} \\ \begin{pmatrix} A_{11}^{-1} & A_{11}^{-1}A_{12} & y_1^0 \\ -A_{21}A_{11}^{-1} & A_{22}-A_{21}A_{11}^{-1}A_{12} & g_2-A_{21}y_1^0-A_{22}y_2^0 \\ -w_1A_{11}^{-1} & w_2-w_1A_{11}^{-1}A_{12} & -wy^0 \end{pmatrix} \begin{matrix} h \\ m-h \\ 1 \end{matrix} \end{array}.$$

Note that y_1^0 is displayed explicitly in this tableau. Also, $y_2^0=0$ since the corresponding nonnegativity constraints are basic and thus binding.

The corresponding dual problem is

$$\begin{aligned} (\text{DLP}) \quad & \max_{x} \quad xr \\ & \text{s.t.} \quad xa^j \leqslant w^j, \quad j=1, \ldots, n, \\ & \qquad xe_i \leqslant 0, \quad i=1, \ldots, m. \end{aligned}$$

To develop a matrix partitioned form of the dual tableau, we proceed as before. Assuming the dual basis consists of h structural constraints and $m-h$ nonnegativity constraints, we have

$$T=(t^1, t^2, \ldots, t^m)=(a^{j_1}, a^{j_2}, \ldots, a^{j_h}, e^{i_{h+1}}, \ldots, e^{i_m}),$$

$$u=(u^1, u^2, \ldots, u^m)=(w^{j_1}, w^{j_2}, \ldots, w^{j_h}, 0, \ldots, 0),$$

so

$$u=(u^1, u^2)=(w^1, 0).$$

The nonbasic constraints are then

$$K=(k^1, k^2, \ldots, k^n)=(e^{i_1}, e^{i_2}, \ldots, e^{i_h}, a^{j_{h+1}}, \ldots, a^{j_n}),$$

$$v=(v^1, v^2, \ldots, v^n)=(0, 0, \ldots, 0, w^{j_{h+1}}, \ldots, w^{j_n}),$$

so $v=(v^1, v^2)=(0, w^2)$.

The matrix partitioned form of the basis is then

$$T=\begin{array}{c} \begin{matrix} h & m-h \end{matrix} \\ \begin{pmatrix} A_{11} & 0 \\ A_{21} & I \end{pmatrix} \begin{matrix} h \\ m-h \end{matrix} \end{array},$$

and with the choice of $Q=T^{-1}$,

$$Q=\begin{array}{c} \begin{matrix} h & m-h \end{matrix} \\ \begin{pmatrix} A_{11}^{-1} & 0 \\ -A_{21}A_{11}^{-1} & I \end{pmatrix} \begin{matrix} h \\ m-h \end{matrix} \end{array},$$

with the remaining constraints forming

$$K=\begin{bmatrix} I & A_{12} \\ 0 & A_{22} \end{bmatrix}.$$

The principal part of the dual tableau is

$$QK=\begin{bmatrix} A_{11}^{-1} & 0 \\ -A_{21}A_{11}^{-1} & I \end{bmatrix}\begin{bmatrix} I & A_{12} \\ 0 & A_{22} \end{bmatrix}=\begin{bmatrix} A_{11}^{-1} & A_{11}^{-1}A_{12} \\ -A_{21}A_{11}^{-1} & A_{22}-A_{21}A_{11}^{-1}A_{12} \end{bmatrix},$$

which we find to be exactly the principal part of the primal tableau, so

$$DP=QK.$$

Thus a single tableau representation supports both primal and dual algorithms.

3. Interpretation of primal and dual forms

We may interpret a primal or dual algorithm as simply different perspectives of this same tableau, wherein a primal algorithm basis change is viewed as exchanging primal constraints and a dual basis change exchanges dual constraints. *The classical Simplex Method may then be interpreted as solving* (PLP) *using the dual perspective*. That the classical Simplex Method is naturally interpreted as a dual algorithm comes as a surprise to the conventionally trained. However, the consequent mathematical insight is compelling, especially in light of the notational simplification and apparent underlying role of A_{11}^{-1}, which we refer to as the *transformation kernel*.

There are several reasons for preferring a primal–dual algorithm to the Simplex Method. From a computational standpoint, because slack variables are carried logically rather than introduced explicitly, we are able to clearly identify the essential information needed to execute the algorithm. The matrix A_{11}^{-1} plays a key role in the calculation of the tableau, and the entire tableau can be constructed from A_{11}^{-1} and original problem data. Since A_{11}^{-1} is a submatrix of the inverse, T^{-1}, used by the Simplex Method, it is smaller and requires fewer arithmetic operations to update than does T^{-1}.

A second advantage of a primal–dual algorithm lies in the flexibility it offers for specialization to particular problem classes or structures. Indeed, it is the special structure and simplicity of the nonnegativity constraints that motivate the development of the algorithm in the first place. It is frequently the case that other special structures can be identified in classes of (PLP). Examples of such structures include simple upper bounds, generalized upper bounds, variable upper bounds, pure and generalized network substructures, etc. Such structure may be "static" in that its nature and dimension remains fixed throughout the solution process, or the structure may be "dynamic" in which case its precise nature and/or dimension may vary as the problem is solved. Some special structures may be more strongly characterized by their column structure and others by their row structure. The primal–dual perspective leads naturally to explanations of the implications of virtually any such problem structure and greatly simplifies the implementation of such a specialization.

When an LP appears as a subproblem in a more sophisticated solution setting (for example, in a mixed integer programming problem or a nonlinear programming problem),

the row/column symmetry of a primal–dual algorithm is of critical importance in specializing the solution approach. The inherent symmetry of such an algorithm permits easy adaptation to branch-and-bound and cutting-plane approaches to mixed integer programming, to column generation settings, as well as to primal and dual decomposition techniques.

We believe the reason for this flexibility offered by the algorithm lies in its more complete mathematical foundation. *There is a natural consistency that arises from the choice of a vector space having the same dimension as the problem variables that is lacking in other approaches.* A natural geometric interpretation of the solution trajectory follows directly from this development. *Incidental issues such as finding an initial basic feasible solution and dealing with degeneracy are resolved constructively in this mathematical framework* (Graves, 1965). Other approaches resort to unnecessarily complicated tangential efforts.

All the research results reported here can be developed, with some effort, in the framework of the classical Simplex Method. However, we choose to present these results in the manner of their development — the mutual primal–dual view presented by Graves.

4. Column and row generation

Rather than maintain a complete tableau *DP*, now consider the generation of just *column* c of this tableau. Rewriting in a manner that highlights our intentions, and labelling row and column partitions for identification

$$DP = \begin{matrix} \\ \text{(i)} \\ \text{(ii)} \end{matrix} \begin{matrix} \begin{matrix} \text{(j)} & \text{(jj)} \end{matrix} \\ \begin{pmatrix} [A_{11}^{-1}] & [A_{11}^{-1}A_{12}] \\ -A_{21}[A_{11}^{-1}] & A_{22} - A_{21}[A_{11}^{-1}A_{12}] \end{pmatrix} \end{matrix}.$$

By properly sequencing our computations we will exploit the fact that region (ii) of a given column is simply a linear combination of terms in region (i) of the same column.

Assume we want to place the current representation of column c into a work array z, which we partition as $z^{\mathrm{T}} = (z_1^{\mathrm{T}}, z_2^{\mathrm{T}})$ to correspond to regions (i) and (ii). Expressed in terms of the transformation kernel A_{11}^{-1}, we compute column c as:
if c is in (j),

$$z_1 = [A_{11}^{-1}]^c,$$

and then

$$z_2 = -A_{21}[z_1];$$

or, if c is in (jj),

$$z_1 = [A_{11}^{-1}(A_{12})^c],$$

and then

$$z_2(A_{22})^c - A_{21}[z_1].$$

Then the current representation of column c is available in $z^{\mathrm{T}} = (z_1^{\mathrm{T}}, z_2^{\mathrm{T}})$.

The computation of *row r* of the tableau proceeds in a similar manner. We now view the principal part of the tableau as

$$DP = \begin{matrix} \\ \text{(i)} \\ \text{(ii)} \end{matrix} \begin{pmatrix} \text{(j)} & \text{(jj)} \\ [A_{11}^{-1}] & [A_{11}^{-1}]A_{12} \\ [-A_{21}A_{11}^{-1}] & A_{22} + [-A_{21}A_{11}^{-1}]A_{12} \end{pmatrix}.$$

If we want to place the current representation of row r in a work array $\tilde{z}$ partitioned conformably with (j) and (jj) as $\tilde{z} = (\tilde{z}_3, \tilde{z}_4)$, we compute:
if row r is in (i),

$$\tilde{z}_3 = [A_{11}^{-1}]_r,$$

and then

$$\tilde{z}_4 = [\tilde{z}_3]A_{12};$$

or, if row r is in (ii),

$$\tilde{z}_3 = [(-A_{21})_r A_{11}^{-1}],$$

and

$$\tilde{z}_4 = (A_{22})_r + [\tilde{z}_3]A_{12},$$

and the current representation of row r is available in $\tilde{z} = (\tilde{z}_3, \tilde{z}_4)$.

We see that in each case calculations proceed by first using a representation of A_{11}^{-1} to compute a portion of the row or column and then using this initial computation and original problem data to compute the remaining part. We will discover that our specializations extend this approach by introducing additional tableau partitions which allow this computational strategy to be applied on a larger scale.

5. Transformation kernel update

The dynamic behavior of A_{11}^{-1} is important. We see from the primal row basis B and nonbasic rows D that the dimension of A_{11}^{-1} corresponds to the number of basic structural constraints, or, equivalently, to the number of nonbasic nonnegativity constraints (recall that if a nonnegativity constraint is nonbasic and thus nonbinding, the corresponding variable may possibly be nonzero). Recalling that our primal view of a basis exchange is as an exchange of constraints between B and D, we see that one of four cases may occur during a pivot:

- A structural constraint enters the basis B and a structural constraint leaves the basis and enters D. Since the number of basic structural constraints (and the number of nonbasic nonnegativity constraints) remains unchanged, the dimension of A_{11}^{-1} is unchanged. A pivot of this type involves a row in region (j) of B and a row in region (ii) of D, and thus it corresponds to a pivot coordinate in the location ((ii), (j)) of the tableau DP.

- A nonnegativity constraint enters the basis and a nonnegativity constraint leaves the basis. Again, the dimension of A_{11}^{-1} remains unchanged. Since this pivot involves a row in region (jj) of B and a row in region (i) of D, the corresponding tableau DP pivot coordinate lies in ((i), (jj)).
- A structural constraint enters the basis and a nonnegativity constraint leaves the basis, and thus the number of basic structural constraints (equivalently, the number of nonbasic nonnegativity constraints) *increases by one*. The dimension of A_{11}^{-1} is increased by one. This corresponds to a pivot coordinate in region ((ii), (jj)) of the tableau DP.
- A nonnegativity constraint enters the basis and a structural constraint leaves the basis, and thus the dimension of A_{11}^{-1} *decreases by one*. The corresponding pivot coordinate in DP is ((i), (j)).

We see that we may exert some influence on the behavior of the dimension of A_{11}^{-1} by our strategy for selecting target exchanges for primal and dual constraints (i.e., our pricing strategy) and through our tie-breaking rules for choosing pivot row/column, and that this dynamism is an inherent feature of an effective algorithm. We have already seen the fundamental importance of the kernel (A_{11}) in our computations. Thus, a successful implementation must manage this dynamic behavior efficiently and reliably.

6. Factorization

The *row-factorized* problem to be considered is

$$
\begin{aligned}
(\mathrm{FLP}) \qquad & \min_{y} \quad wy \\
& \text{s.t.} \quad Ey \leqslant r \qquad \text{(explicit constraints)}, \\
& \qquad\quad Fy \leqslant b \qquad \text{(factored constraints)}, \\
& \qquad\quad -Iy \leqslant 0 \qquad \text{(nonnegativity constraints)},
\end{aligned}
$$

where y is an n-vector of decision variables, w a vector of cost coefficients, E a matrix of constraint coefficients for "explicit" constraints with right-hand side m-vector, r, F a matrix of constraint coefficients for "factored" constraints with right-hand side p-vector b, and $-I$ the negative of the identity matrix. In this general development, we refer to the F-type constraints as "factored" only to distinguish them from the "explicit" E-type constraints, and assume nothing about their structure. Not until our specializations later will we impose special structure on F, and the structures we will consider may permit the representation of the F-type constraints without the inversion of a matrix. We will see that this approach is centered around handling the part of the basis corresponding to the E-type constraints explicitly while factoring the portion of the basis corresponding to the F-type constraints. The notation is chosen to suggest this idea.

Recall that a basis for the primal algorithm consists of n linearly independent rows from the constraint matrix when it is assumed to include both structural (explicit and factored)

and nonnegativity constraints. Assume that the current row basis consists of k rows from E, l rows from F and $n-(k+l)$ rows from $-I$. Repeating our notation

$$B=\begin{pmatrix} A_{11} & A_{12} \\ 0 & -I \end{pmatrix}\begin{matrix} k+l \\ n-(k+l) \end{matrix}\quad \begin{matrix} k+l & n-(k+l) \end{matrix},$$

where $[A_{11}\ A_{12}]$ includes all basic structural rows, from both E and F.

We will ultimately be interested in isolating the effect of each type of structural constraint algebraically in the factored tableau, and thus we require greater resolution in our factored basis. Introducing obvious notation, we have

$$[A_{11}\ A_{12}]=\begin{matrix} \\ k \\ l \end{matrix}\begin{matrix} \begin{matrix} k & l & n-(k+l) \end{matrix} \\ \begin{pmatrix} E_{11} & E_{12} & E_{13} \\ F_{21} & F_{22} & F_{23} \end{pmatrix} \end{matrix},$$

where the kernel of dimension $k+l$ is given by

$$[A_{11}]=\begin{bmatrix} E_{11} & E_{12} \\ F_{21} & F_{22} \end{bmatrix}.$$

Because A_{11} is a basis for $\mathbb{R}^{k+l}$ it follows that it is always possible to identify among the columns of $[F_{21}\ F_{22}]$ a nonsingular submatrix F_{22} of dimension l, since otherwise the rank of $[F_{21}\ F_{22}]$ is at most $l-1$ and thus the rank of A_{11} is at most $k+l-1$, or equivalently the rank of B is at most $n-1$. We will later see that one of the important implementation challenges is the task of efficiently managing the structure and nonsingularity of F_{22}.

The full factored row basis is then

$$B=\begin{matrix} \\ \text{(j)} \\ \text{(jj)} \\ \text{(jjj)} \end{matrix}\begin{matrix} \begin{matrix} k & l & n-(k+l) \end{matrix} \\ \begin{pmatrix} E_{11} & E_{12} & E_{13} \\ F_{21} & F_{22} & F_{23} \\ 0 & 0 & -I \end{pmatrix} \end{matrix}\begin{matrix} k \\ l \\ n-(k+l) \end{matrix}.$$

Introducing the notation

$$\tilde{A}_{11}\equiv E_{11}-E_{12}F_{22}^{-1}F_{21},\qquad \tilde{A}_{13}\equiv E_{13}-E_{12}F_{22}^{-1}F_{23},$$

where $\tilde{A}_{11}$ is the Schur complement, or Gauss transform, of F_{22} in A_{11} (e.g., Golub and Van Loan, 1983), we can write its inverse as

$$B^{-1}=\begin{bmatrix} \tilde{A}_{11}^{-1} & -\tilde{A}_{11}^{-1}E_{12}F_{22}^{-1} & \tilde{A}_{11}^{-1}\tilde{A}_{13} \\ -F_{22}^{-1}F_{21}\tilde{A}_{11}^{-1} & (I+F_{22}^{-1}F_{21}\tilde{A}_{11}^{-1}E_{12})F_{22}^{-1} & F_{22}^{-1}(F_{23}-F_{21}\tilde{A}_{11}^{-1}\tilde{A}_{13}) \\ 0 & 0 & -I \end{bmatrix}.$$

Grouping the coefficients of the nonbasic constraints and applying the same column ordering yields

$$D = \begin{array}{c} \\ \text{(i)} \\ \text{(ii)} \\ \text{(iii)} \\ \text{(iv)} \end{array} \begin{array}{c} \begin{array}{ccc} k & l & n-(k+l) \end{array} \\ \begin{pmatrix} -I & 0 & 0 \\ 0 & -I & 0 \\ E_{31} & E_{32} & E_{33} \\ F_{41} & F_{42} & F_{43} \end{pmatrix} \end{array} \begin{array}{l} k \\ l \\ m-k \\ p-l \end{array}.$$

The principal part of the factored tableau is DP, where $P = -B^{-1}$ is the conjugate row basis. With the additional notation

$$\tilde{A}_{31} \equiv E_{31} - E_{32}F_{22}^{-1}F_{21}, \qquad \tilde{A}_{33} \equiv E_{33} - E_{32}F_{22}^{-1}F_{23},$$
$$\tilde{F}_{41} \equiv F_{41} - F_{42}F_{22}^{-1}F_{21}, \qquad \tilde{F}_{43} \equiv F_{43} - F_{42}F_{22}^{-1}F_{23},$$

the principal part of the factored tableau is

$$DP = \begin{array}{c} \\ \text{(i)} \\ \text{(ii)} \\ \text{(iii)} \\ \text{(iv)} \end{array} \begin{array}{c} \begin{array}{ccc} \text{(j)} & \text{(jj)} & \text{(jjj)} \end{array} \\ \begin{pmatrix} \tilde{A}_{11}^{-1} & -\tilde{A}_{11}^{-1}E_{12}F_{22}^{-1} & \tilde{A}_{11}^{-1}\tilde{A}_{13} \\ -F_{22}^{-1}F_{21}\tilde{A}_{11}^{-1} & (I+F_{22}^{-1}F_{21}\tilde{A}_{11}^{-1}E_{12})F_{22}^{-1} & F_{22}^{-1}(F_{23}-F_{21}\tilde{A}_{11}^{-1}\tilde{A}_{13}) \\ -\tilde{A}_{31}\tilde{A}_{11}^{-1} & (\tilde{A}_{31}\tilde{A}_{11}^{-1}E_{12}-E_{32})F_{22}^{-1} & \tilde{A}_{33}-\tilde{A}_{31}\tilde{A}_{11}^{-1}\tilde{A}_{13} \\ -\tilde{F}_{41}\tilde{A}_{11}^{-1} & (\tilde{F}_{41}\tilde{A}_{11}^{-1}E_{12}-F_{42})F_{22}^{-1} & \tilde{F}_{43}-\tilde{F}_{41}\tilde{A}_{11}^{-1}\tilde{A}_{13} \end{pmatrix} \end{array}.$$

Partitioning $w = (w_1, w_2, w_3, w_4)$, $r^{\mathrm{T}} = (r_1^{\mathrm{T}}, r_2^{\mathrm{T}})$ and $b^{\mathrm{T}} = (b_1^{\mathrm{T}}, b_2^{\mathrm{T}})$ and introducing the notation

$$\tilde{w}_2 \equiv w_2 - w_1F_{22}^{-1}F_{21}, \qquad \tilde{w}_3 \equiv w_3 - w_2F_{22}^{-1}F_{23},$$
$$\tilde{b}_2 \equiv b_2 - F_{41}F_{22}^{-1}b_1, \qquad \tilde{r}_1 \equiv r_1 - E_{11}F_{22}^{-1}b_1, \qquad \tilde{r}_2 \equiv r_2 - E_{21}F_{22}^{-1}b_1,$$

the complete factored tableau is

$$\begin{bmatrix} \tilde{A}_{11}^{-1} & -\tilde{A}_{11}^{-1}E_{12}F_{22}^{-1} & \tilde{A}_{11}^{-1}\tilde{A}_{13} & \tilde{A}_{11}^{-1}\tilde{r}_1 \\ -F_{22}^{-1}F_{21}\tilde{A}_{11}^{-1} & (I+F_{22}^{-1}F_{21}\tilde{A}_{11}^{-1}E_{12})F_{22}^{-1} & F_{22}^{-1}(F_{23}-F_{21}\tilde{A}_{11}^{-1}\tilde{A}_{13}) & F_{22}^{-1}(b_1-F_{21}\tilde{A}_{11}^{-1}\tilde{r}_1) \\ -\tilde{A}_{31}\tilde{A}_{11}^{-1} & (\tilde{A}_{31}\tilde{A}_{11}^{-1}E_{12}-E_{32})F_{22}^{-1} & \tilde{A}_{33}-\tilde{A}_{31}\tilde{A}_{11}^{-1}\tilde{A}_{13} & \tilde{r}_2-\tilde{A}_{31}\tilde{A}_{11}^{-1}\tilde{r}_1 \\ -\tilde{F}_{41}\tilde{A}_{11}^{-1} & (\tilde{F}_{41}\tilde{A}_{11}^{-1}E_{12}-F_{42})F_{22}^{-1} & \tilde{F}_{43}-\tilde{F}_{41}\tilde{A}_{11}^{-1}\tilde{A}_{13} & \tilde{b}_2-\tilde{F}_{41}\tilde{A}_{11}^{-1}\tilde{r}_1 \\ -w_2\tilde{A}_{11}^{-1} & (\tilde{w}_2\tilde{A}_{11}^{-1}E_{12}-w_1)F_{22}^{-1} & \tilde{w}_3-\tilde{w}_2\tilde{A}_{11}^{-1}\tilde{A}_{13} & w_1F_{22}^{-1}b_1+\tilde{w}_2\tilde{A}_{11}^{-1}\tilde{r}_1 \end{bmatrix}$$

We see that with knowledge of the current factorization, we can construct the entire tableau from F_{22}^{-1}, $\tilde{A}_{11}^{-1}$ and the original problem data. The dimension of F_{22}^{-1} is equal to the number of F-type constraints that are currently basic, and thus can be at most p. The dimension of $\tilde{A}_{11}^{-1}$ is equal to the number of E-type constraints that are currently basic, and thus cannot exceed m. We call $\tilde{A}_{11}^{-1}$ the *explicit transformation kernel* and F_{22}^{-1} the *factored transformation kernel.*

7. Factored column and row generation

Consider generation of column c from the principal part of the tableau DP. Rewriting in a manner that highlights our intentions

$$DP = \begin{array}{c} \\ \text{(i)} \\ \text{(ii)} \\ \text{(iii)} \\ \text{(iv)} \end{array} \begin{pmatrix} \text{(j)} & \text{(jj)} & \text{(jjj)} \\ [\tilde{A}_{11}^{-1}] & [-\tilde{A}_{11}^{-1}E_{12}F_{22}^{-1}] & [\tilde{A}_{11}^{-1}\tilde{A}_{13}] \\ \{-F_{22}^{-1}F_{21}[\ ]\} & \{F_{22}^{-1}-F_{22}^{-1}F_{21}[\ ]\} & \{F_{22}^{-1}F_{23}-F_{22}^{-1}F_{21}[\ ]\} \\ -E_{31}[\ ]-E_{32}\{\ \} & -E_{31}[\ ]-E_{32}\{\ \} & E_{33}-E_{31}[\ ]-E_{32}\{\ \} \\ -F_{41}[\ ]-F_{42}\{\ \} & -F_{41}[\ ]-F_{42}\{\ \} & F_{43}-F_{41}[\ ]-F_{42}\{\ \} \end{pmatrix}$$

where $\tilde{A}_{13}$ is defined as before, and the brackets "[ ]" and "{ }" contain terms common to but displayed only once for each column.

Assume we want to place column c into work array z. We partition z conformably as $z^{\mathrm{T}}=(z_1^{\mathrm{T}}, z_2^{\mathrm{T}}, z_3^{\mathrm{T}}, z_4^{\mathrm{T}})$, refer similarly to components of unit vector e^c, and employ a $(m+p)$-vector work array $\bar{z}$. The notation "$\leftarrow$" denotes simple assignment, "$=$" indicates that a set of factored equations must be solved, and "$\Rightarrow$" explains the corresponding result.

If column c is in (j),

$$z_1 \leftarrow [\tilde{A}_{11}^{-1}]^c,$$

and then

$$\bar{z} \leftarrow -F_{21}[z_1],$$

$$F_{22}z_2 = \bar{z} \;\Rightarrow\; z_2 \leftarrow \{-F_{22}^{-1}F_{21}[z_1]\},$$

then

$$z_3 \leftarrow -E_{31}[z_1] - E_{32}\{z_2\},$$

and finally

$$z_4 \leftarrow -F_{41}[z_1] - F_{42}\{z_2\};$$

if c is in (jj), solve

$$F_{22}z_2 = -(e_2)^c \;\Rightarrow\; z_2 \leftarrow -(F_{22}^{-1})^c,$$

$$\bar{z} \leftarrow E_{12}z_2 \;\Rightarrow\; \bar{z} \leftarrow -(E_{12}F_{22}^{-1})^c,$$

$$z_1 \leftarrow \tilde{A}_{11}^{-1}\bar{z} \;\Rightarrow\; z_1 \leftarrow [-\tilde{A}_{11}^{-1}E_{12}F_{22}^{-1}]^c,$$

then

$$\bar{z} \leftarrow (e_2)^c - F_{21}[z_1],$$

$$F_{22}z_2 = \bar{z} \;\Rightarrow\; z_2 \leftarrow \{(F_{22}^{-1})^c - F_{22}^{-1}F_{21}[z_1]\},$$

and finally

$$z_3 \leftarrow -E_{31}[z_1] - E_{32}\{z_2\},$$

$$z_4 \leftarrow -F_{41}[z_1] - F_{42}\{z_2\};$$

if c is in (jjj),

$$\bar{z} \leftarrow (F_{23})^c,$$

$$F_{22}z_2 = \bar{z} \Rightarrow z_2 \leftarrow F_{22}^{-1}(F_{23})^c,$$

$$\bar{z} \leftarrow (E_{13})^c - E_{12}z_2 \Rightarrow \bar{z} \leftarrow (\tilde{A}_{13})^c,$$

$$z_1 \leftarrow \tilde{A}_{11}^{-1}\bar{z} \Rightarrow z_1 \leftarrow [\tilde{A}_{11}^{-1}\tilde{A}_{13}]^c,$$

then

$$\bar{z} \leftarrow (F_{23})^c - F_{21}z_1,$$

$$F_{22}z_2 = \bar{z} \Rightarrow z_2 \leftarrow \{F_{22}^{-1}(F_{23})^c - F_{22}^{-1}F_{21}[z_1]\},$$

and finally

$$z_3 \leftarrow (E_{33})^c - E_{31}[z_1] - E_{32}\{z_2\},$$

$$z_4 \leftarrow (F_{43})^c - F_{41}[z_1] - E_{42}\{z_2\}.$$

Similarly, row r can be generated. The principal part of the tableau is now viewed as

$$DP = \begin{array}{c} \\ \text{(i)} \\ \text{(ii)} \\ \text{(iii)} \\ \text{(iv)} \end{array} \begin{pmatrix} \text{(j)} & \text{(jj)} & \text{(jjj)} \\ [\tilde{A}_{11}^{-1}] & \{-[\ ]E_{12}F_{22}^{-1}\} & [\ ]E_{13} + \{\ \}F_{23} \\ [-F_{22}^{-1}F_{21}\tilde{A}_{11}^{-1}] & \{F_{22}^{-1} - [\ ]E_{12}F_{22}^{-1}\} & [\ ]E_{13} + \{\ \}F_{23} \\ [-\tilde{A}_{31}\tilde{A}_{11}^{-1}] & \{(-E_{32} - [\ ]E_{12})F_{22}^{-1}\} & E_{33} + [\ ]E_{13} + \{\ \}F_{23} \\ [-\tilde{F}_{41}\tilde{A}_{11}^{-1}] & \{(-F_{42} - [\ ]E_{12})F_{22}^{-1}\} & F_{43} + [\ ]E_{13} + \{\ \}F_{23} \end{pmatrix},$$

where $\tilde{A}_{31}$ and $\tilde{F}_{41}$ are defined as before, and the brackets "[ ]" and "{ }" contain terms common to but displayed only once for each row.

Assume we want to place row r into work array $\bar{z}$. We partition $\bar{z}$ conformably as $\bar{z} = (\bar{z}_5, \bar{z}_6, \bar{z}_7)$, refer similarly to components of unit vector e_r, and employ a $(m+p)$-vector work array $\bar{z}$.

If row r is in (i),

$$\bar{z}_5 \leftarrow [\tilde{A}_{11}^{-1}]_r,$$

then

$$\bar{z} \leftarrow -\bar{z}_5E_{12} \Rightarrow \bar{z} \leftarrow -[\tilde{A}_{11}^{-1}]_rE_{12},$$

$$\bar{z}_6F_{22} = \bar{z} \Rightarrow \bar{z}_6 \leftarrow \{-[\tilde{A}_{11}^{-1}]_rE_{12}F_{22}^{-1}\},$$

and finally

$$\bar{z} \leftarrow [\bar{z}_5]E_{13} + \{\bar{z}_6\}F_{23};$$

if r is in (ii), solve

$$\bar{z}_6 F_{22} = -(e_6)_r \;\Rightarrow\; \bar{z}_6 \leftarrow -(F_{22}^{-1})_r,$$

$$\bar{z} \leftarrow \bar{z}_6 F_{21} \;\Rightarrow\; \bar{z} \leftarrow -(F_{22}^{-1})_r F_{21},$$

$$\bar{z}_5 \leftarrow \bar{z}\tilde{A}_{11}^{-1} \;\Rightarrow\; \bar{z}_5 \leftarrow [-F_{22}^{-1}F_{21}\tilde{A}_{11}^{-1}]_r,$$

then

$$\bar{z} \leftarrow (e_6)_r - [\bar{z}_5]E_{12},$$

$$\bar{z}_6 F_{22} = \bar{z} \;\Rightarrow\; \bar{z}_6 \leftarrow \{(F_{22}^{-1})_r - [\bar{z}_5]E_{12}F_{22}^{-1}\},$$

and finally

$$\bar{z}_7 \leftarrow [\bar{z}_5]E_{13} + \{\bar{z}_6\}F_{23};$$

if r is in (iii),

$$\bar{z} \leftarrow -(E_{32})_r,$$

$$\bar{z}_6 F_{22} = \bar{z} \;\Rightarrow\; \bar{z}_6 \leftarrow -(E_{32})_r F_{22}^{-1},$$

$$\bar{z} \leftarrow (E_{31})_r + \bar{z}_6 F_{21} \;\Rightarrow\; \bar{z} \leftarrow (\tilde{A}_{31})_r,$$

$$\bar{z}_5 \leftarrow [\bar{z}\tilde{A}_{11}^{-1}] \;\Rightarrow\; \bar{z}_5 \leftarrow [-\tilde{A}_{31}\tilde{A}_{11}^{-1}]_r,$$

then

$$\bar{z} \leftarrow -(E_{32})_r - [\bar{z}_5]E_{12},$$

$$\bar{z}_6 F_{22} = \bar{z} \;\Rightarrow\; \bar{z}_6 \leftarrow \{(-(E_{32})_r - [\bar{z}_5]E_{12})F_{22}^{-1}\},$$

and finally

$$\bar{z}_7 \leftarrow (E_{33})_r + [\bar{z}_5]E_{13} + \{\bar{z}_6\}F_{23};$$

if r is in (iv),

$$\bar{z} \leftarrow -(F_{42})_r,$$

$$\bar{z}_6 F_{22} = \bar{z} \;\Rightarrow\; \bar{z}_6 \leftarrow -(F_{42})_r F_{22}^{-1},$$

$$\bar{z} \leftarrow (F_{41})_r - \bar{z}_6 F_{21} \;\Rightarrow\; \bar{z} \leftarrow (\tilde{F}_{41})_r,$$

$$\bar{z}_5 \leftarrow [\bar{z}\tilde{A}_{11}^{-1}] \;\Rightarrow\; \bar{z}_5 \leftarrow [-\tilde{F}_{41}\tilde{A}_{11}^{-1}]_r,$$

then

$$\bar{z} \leftarrow -(F_{42})_r - [\bar{z}_5]E_{12},$$

$$\bar{z}_6 F_{22} = \bar{z} \;\Rightarrow\; \bar{z}_6 \leftarrow \{(-(F_{42})_r - [\bar{z}_5]E_{12})F_{22}^{-1}\},$$

and finally

$$\tilde{z}_7 \leftarrow (F_{43})_r + [\tilde{z}_5]E_{13} + \{\tilde{z}_6\}F_{23}.$$

8. The complete algorithm

The complete algorithm is described in terms of abstract functions which operate on fundamental data structures.

Tableau management requires two index maps: one yielding the intrinsic coordinate in the principal part of the tableau for each original, extrinsic problem row or column, and the other its inverse map. Intrinsic arguments are shown in lower case, and extrinsic in upper case. **Index_Exchange**(index1,index2) updates these maps for the exchange of a pair of tableau coordinates index1 and index2.

The tableau regions are successive partitions of indices:

Tableau region	ENDING INDEX	Contents of region
(i)	MEC	basic **C**olumns solving **E**xplicit rows
(ii)	MFC	basic **C**olumns solving **F**actored rows
(iii)	MER	nonbasic **E**xplicit **R**ows
(iv)	m	nonbasic **F**actored **R**ows
(j)	NER	basic **E**xplicit **R**ows
(jj)	NFR	basic **F**actored **R**ows
(jjj)	$m+n$	nonbasic **C**olumns

Increment(ending_index) and **Decrement**(ending_index) are functions to modify these ending indices.

Generate_Row(row) and **Generate_Column**(column) place numeric values of a tableau row or column in ROWCOL( ), which is commensurate with the tableau dimensions.

Using ROWCOL( ), **Update_Rim**(row,col) maintains current numeric values of the right-hand side and bottom row of the tableau in RIM( ).

The explicit transformation kernel, $\tilde{A}_{11}^{-1}$, is operated on by functions using ROWCOL( ):

Add_E-Kernel_Row(ROW),
Add_E-Kernel_Column(COLUMN),
Delete_E-Kernel_Row(ROW),
Delete_E-Kernel_Column(COLUMN),
Replace_E-Kernel_Row(REPLACED_ROW,REPLACING_ROW), and
Update_Explicit_Transformation_Kernel, the pivotal update.

Table 8.1
Secondary and tertiary tableau exchanges

	(j)	(jj)	(jjj)
	Delete_E-Kernel_Column(ROW)	**Index_Exchange**(col,NFR)	
	Index_Exchange(col,NER)	**Decrement**(NFR);	
	Index_Exchange(NER,NFR)	**Find_F-Kernel_Column_to_Remove**(COL,KOU)	
	Decrement(NER)	**Generate_Row**(kou)	
(i)	**Decrement**(NFR);	**Replace_E-Kernel_Row**(ROW,KOU)	(no more exchanges)
	Delete_E-Kernel_Row(COL)	**Index_Exchange**(row,kou)	
	Index_Exchange(row,MEC)	**Index_Exchange**(row,MFC)	
	Index_Exchange(MEC,MFC)	**Index_Exchange**(MFC,MER)	
	Decrement(MEC)	**Decrement**(MFC)	
	Decrement(MFC)	**Decrement**(MER)	
	Delete_E-Kernel_Column(ROW)	**Index_Exchange**(col,NFR)	(no more column exchanges);
	Index_Exchange(col,NER)	**Decrement**(NFR);	
	Index_Exchange(NER,NFR)	**IF (Factored_Kernel_Singular**(COL,ROW)) **THEN**	**IF (Factored_Kernel_Singular**(ROW,COL)) **THEN**
	Decrement(NER)	**Find_F-Kernel_Column_to_Remove**(COL,KOU)	**Find_E-Kernel_Column_for_Key**(ROW,KIN)
	Decrement(NFR);	**Find_E-Kernel_Column_for_Key**(ROW,KIN)	**Replace_E-Kernel_Row**(KIN,ROW)
(ii)	**Find_E-Kernel_Column_for_Key**(ROW,KIN)	**Generate_Row**(kou)	**Index_Exchange**(kin,row)
	Delete_E-Kernel_Row(KIN)	**Replace_E-Kernel_Row**(KIN,KOU)	**ENDIF**
	Index_Exchange(kin,MEC)	**Index_Exchange**(kin,kou)	
	Index_Exchange(row,MFC)	**ENDIF**	
	Decrement(MEC)	**Index_Exchange**(row,MFC)	
	Decrement(MFC)	**Index_Exchange**(MFC,MER)	
		Decrement(MFC)	
		Decrement(MER)	

		Add_E-Kernel_Column(COL)	**Add_E-Kernel_Column**(COL)
		Increment(NER);	**Increment**(NER)
		Index_Exchange(col,NER)	**Increment**(NFR)
		Find_F-Kernel_Column_to_Remove(COL,KOU)	**Index_Exchange**(col,NFR)
(iii)	(no more exchanges)	**Generate_Row**(kou)	**Index_Exchange**(NFR,NER);
		Add_E-Kernel_Row(KOU)	**Add_E-Kernel_Row**(ROW)
		Index_Exchange(kou,MEC)	**Increment**(MEC)
		Increment(MEC)	**Increment**(MFC)
		Index_Exchange(row,MER)	**Index_Exchange**(row,MEC)
		Decrement(MER)	**Index_Exchange**(MEC,MFC)
	Delete_E-Kernel_Column(ROW)	(no more column exchanges);	**Increment**(NFR)
	Index_Exchange(col,NER)		**Index_Exchange**(col,NFR);
	Decrement(NER);	**IF (Factored_Kernel_Singular**(COL,ROW)) **THEN**	**IF (Factored_Kernel_Singular**(COL,ROW)) **THEN**
	Delete_E-Kernel_Row(COL)	**Find_F-Kernel_Column_to_Remove**(COL,KOU)	**Find_E-Kernel_Column_for_Key**(ROW,KIN)
(iv)	**Find_E-Kernel_Column_for_Key**(ROW,KIN)	**Generate_Row**(kou)	**Replace_E-Kernel_Row**(KIN,ROW)
	Index_Exchange(kin,MEC)	**Find_E-Kernel_Column_for_Key**(ROW,KIN)	**Index_Exchange**(kin,row)
	Decrement(MEC)	**Replace_E-Kernel_Row**(KIN,KOU)	**ENDIF**
	Increment(MER)	**Index_Exchange**(kin,kou)	**Increment**(MFC)
	Index_Exchange(row,MER)	**ENDIF**	**Increment**(MER)
			Index_Exchange(row,MER)
			Index_Exchange(MER,MFC)

A pivot in tableau row "row" and column "col" is associated with problem row or column indices ROW and COL, and begins with **Generate_Row**(row), **Generate_Column**(col), **Update_Rim**(row,col), and primary **Index_Exchange**(row,col). Next, depending upon the tableau region, secondary and even tertiary exchanges shown above may be required to preserve the factorization and the tableau structure so the pivot can then be completed with **Update_Factored_Kernel**(ROW,COL) and **Update_Explicit_Transformation_Kernel**.

$\tilde{A}_{11}^{-1}$ can be represented in any way that suits the implementer and efficiently supports these functions. Generally, we find $\tilde{A}_{11}^{-1}$ to be relatively sparse, and report here results using an array with an entry-point for each inverse row and column which accesses a stack of orthogonally-linked nonzero inverse elements. We have also implemented a dense vector-processor version, and the design issues for LU-based schemes are given by Olson (1989).

The factored kernel, F_{22}, is manipulated with:

Factored_Kernel_Singular(ROW,COLUMN), a Boolean function,
Find_E-Kernel_Column_for_Key(ROW,COLUMN), a column from region (i),
Find_F-Kernel_Column_to_Remove(ROW,COLUMN), a column from region (ii),
and
Update_Factored_Kernel(ROW,COLUMN), the pivotal update.

The complete abstract algorithm is:

Step 0. Initialize.
Step 1. Select primal or dual algorithm.
Step 2. Select a primal (col) or dual (row) violation;
STOP if current solution is terminal, or
Generate_Column(col) or **Generate_Row**(row).
Step 3. By ratio test with RIM( ) and ROWCOL( ),
select a (row) or (col) pivot coordinate;
STOP if current solution is terminal, or
Generate_Row(row) or **Generate_Column**(col).
Step 4. **Update_Rim**(row,col),
primary **Index_Exchange**(row,col),
perform secondary and tertiary exchanges (Table 8.1),
Update_Factored_Kernel(ROW,COL),
Update_Explicit_Transformation_Kernel,
Go to Step 1.

Functions for maintaining the factored kernel vary with the factorization, and a good implementation will exploit these differences. However, a general specification will suffice for all factorizations here.

We are exclusively interested in performing two fundamental operations:

1. Solving factored equations of the form

$$F_{22} z_2 = b_2$$

and

$$z_2^{\mathrm{T}} F_{22} = \tilde{b}_2^{\mathrm{T}}$$

where z_2 and $\tilde{z}_2$ are unknown and b_2 and $\tilde{b}_2$ are rational (not necessarily integer).

2. Restoring $\hat{F}_{22}$ to the desired form of F_{22} which makes the factored equations above easy to solve, where $\hat{F}_{22}$ results from inflicting F_{22} with a column exchange, a column and row deletion, a column and row addition, or a row exchange.

Factored_Kernel_Singular(LABEL1,LABEL2) predicts whether $\hat{F}_{22}$ will be singular if:

1. LABEL1 gives a basic column in a basic factored ROW for which COL = LABEL2 would be exchanged;

2. LABEL1 is a basic factored ROW solved by basic column COL = LABEL2, both of which would be removed from the factored kernel;

3. LABEL1 is a nonbasic factored ROW, which would be added to the factored kernel with column COL = LABEL2; or

4. LABEL1 is a basic factored ROW which would be replaced by some other row LABEL2.

The first case, a column exchange, is equivalent to asking whether solving $F_{22}z_2 = b_2$ with b_2 equal to region (ii) of the proposed entering column COL, yields a nonzero term associated with the basic factored row in z_2(row). This is easy to answer for the factorizations we discuss. For instance, Brown and McBride (1984) show that back-solving a "nearly-triangulated generalized network" basis F_{22} only as far as ROW will suffice, and that this can be implemented as a traversal of the "backpaths" of the (zero, one, or two) coefficients in b_2 for the column now basic in ROW. If numerical precision is an issue, z_2(row) can be computed during this search and tested for significance.

The second case, a row and column deletion, is trivial because the resulting F_{22} will always be nonsingular.

The third case, a row and column addition, can be answered by adding ROW and COL to F_{22} without disturbing its desirable special structure, thus creating an instance of case one. For instance, placing ROW first, and COL last in F_{22} suffices for all factorizations discussed here.

The fourth case, a row exchange, can be answered by applying the second case, deleting ROW and its basic column, and then (perhaps repeatedly) applying the third case, adding the row with index COL and any column which would yield a nonsingular F_{22}.

Find_F-Kernel_Column_to_Remove(ROW,COL) given basic factored row ROW, returns its basic, or "key" column COL.

Find_E-Kernel_Column_for_Key(LABEL,COL), given either a nonbasic factored ROW = LABEL, or a basic column LABEL solving a basic factored ROW, searches for an acceptable basic column COL in Explicit Rows (region (i)) using **Factored_Kernel_Singular**(ROW,COL).

Update_Factored_Kernel(LABEL1,LABEL2) restores $\hat{F}_{22}$ to the desired form of F_{22}, with a possible increase or decrease in dimension. Following the case-by-case scheme of **Factored_Kernel_Singular**, we pre- and post-process the factored basis representation to

permit use of a single, static factorization update of conventional design. Olson (1989) pursues this in considerable detail.

Table 8.2 displays supporting data structures for these functions.

9. Computational experience

The factorization methods introduced here have been implemented and used to solve a variety of existing models provided by our colleagues. With their help, we have extracted suggested problem instances from a diversity of decision support systems on host computers ranging from mainframes to micro-computers. Because our goal is to test factorization technology in isolation, the results reported here are achieved without benefit of any model-specific knowledge or tuning.

However, we also seek to develop effective modeling tools for customized use in developing and refining new models. To this end, we have greatly benefited from the experience and advice of our colleagues, and we include some discussion of modeler guidance and insight along with the numerical results.

Each model is introduced below by a short synopsis. Multiple instances of some models

Table 8.2
Factorization algorithm data structures

Factorizations	Data Structure	Size	Use
GUB,PN,GN	RIM( )	$m+n$	current tableau right-hand side, bottom row
	ROWCOL( )	$m+n$	current tableau row and column
	MSKRC( )	$m+n$	logical mask true for corresponding nonzero in ROWCOL( ), false otherwise
	LQRC( )	$m+n$	LIFO queues of nonzero row and column coordinates in ROWCOL( )
	KEY( )	$m+n$	basic column in basic factored row, and vice versa
PN,GN	WORK( )	p	values for b_2 or $\tilde{b}_2$ in basic factored equations
	MSKWK( )	p	logical mask for nonzero row and column coordinates in WORK( )
	LQWK( )	p	LIFO queue of nonzero coordinates in WORK( )
	PO( )	p	next basic factored row in pre-order
	P( )	p	off-diagonal row with nonzero factored coefficient
	D( )	p	depth, remaining back-substitution path length in factored component
GN	VGN( )	p	generalized network cycle factors
	JMUL( )	n	ratio of generalized network coefficients in each column

Generalized upper bound (GUB), pure network (PN), and generalized network (GN) factorizations respectively require more data structures to support kernel factorization. For each factorization, F_{22} is maintained in some partial ordering of rows, and of columns — a signed identity for GUB, upper-triangular for PN (e.g., Bradley, Brown, and Graves, 1977), and nearly-upper-triangular for GN (e.g., Brown and McBride, 1984). Direct solution of factored kernel equations $F_{22}z_2 = b_2$ and $\tilde{z}_2^{\mathrm{T}} F_{22} = \tilde{b}_2^{\mathrm{T}}$ is performed alternately using the data structures shown.

are reported where diversity of size, structure, and taxonomy have proven interesting to the modelers. For those models that employ nonlinear, mixed integer, or decomposition features, we report solution statistics for the initial linear program.

• *GTE.* The seven Telephone Operating Companies within GTE have adopted an integrated business system called Capital Program Management System (CPMS) to guide their 3 billion dollar per year capital planning. The system includes a large scale mixed integer programming optimization system that optimizes the critical economic tradeoffs between maximizing the long-term budget value of the firm's equity and satisfying shorter-term financial constraints, resource limitations and service objectives. Investment opportunities for the next 5 years are modeled as 0–1 variables with alternative implementations for each. The objective is to maximize the net present value of the capital portfolio. There are financial constraints on capital, internally generated funds, net income to common, and limits on resources such as labor hours, lines installed, etc. There are also constraints that enforce logical relationships among opportunities (such as, if choose A then must choose B). See Bradley (1986).

• *INVEST.* Capital allocation and project selection for Mobil Oil Corporation are modeled as a two-stage multi-year nonlinear capital budgeting problem with over 40 000 integer variables. A master problem allocates capital among markets over a multi-year horizon considering the estimated nonlinear effects on sales of concentrated marketing investments. The instance reported here is a mixed-integer linear program subproblem of the two-stage model which, given these annual capital expenditure limits for a market, selects particular alternate investments. Such subproblems are easy to solve, and optimality is achieved with a single iteration of the nonlinear master problem. See Harrison, Bradley and Brown (1989).

• *TANKER.* A crude oil tanker scheduling problem faced by a major oil company is solved using an elastic set partitioning model. The model takes into account all fleet cost components, including the opportunity cost of ship time, port and canal charges, demurrage and bunker fuel. The model determines optimal speeds for the ships and the best routing of ballast (empty) legs, as well as which cargoes to load on controlled ships and which to spot charter. All feasible schedules are generated, the cost of each is accurately determined and the best set of schedules is selected. See Brown, Graves and Ronen (1987).

• *HFDF.* A large-scale elastic set partitioning model used to assign frequencies for a network of high frequency direction finding receivers. See Brown, Drake, Marsh, and Washburn (1990).

• *GAS.* A multi-time period strategic model for use by natural gas utilities for determining optimal contract levels for gas purchase, storage and transmission. An underlying generalized network flow model represents gas being bought, stored, shipped and consumed over a multi-year horizon, typically at a monthly level of detail. Constraints and variables are added to handle variable maximum and minimum purchase levels, variable leased or constructed storage and variable transmission capacities. An integrated parallel model incorporates the peak requirements necessary on some days during cold winter months. This model has been used by a number of utilities including Southwest Gas Corporation and

Questar Pipeline Corporation to plan operations and to justify such plans to regulatory agencies. See Avery, Brown, Rosenkranz and Wood (1992).

• *KELLOGG.* A multi-time period, multi-plant production/inventory/transshipment linear program for Kellogg cereals. The model guides weekly processing, packaging and shipping decisions. Production consists of two stages: processing lines produce basic products which are then packaged on packaging lines into different-sized containers to yield finished products. Processing lines produce a subset of the basic products and have limited capacity with overtime charges for weekend shifts. Packaging lines are analogous. In-house inventory capacity is limited although outside storage is available to additional cost. Interplant shipments of finished products are made by rail or truck. See Wood (1989).

• *ODS.* A commonly occurring problem in distribution system design is the optimal location of intermediate distribution facilities between plants and customers. A multicommodity capacitated single-period version of this problem is formulated as a mixed integer linear program. A solution technique based on Benders Decomposition is developed. ...An essentially optimal solution is found and proven with a surprisingly small number of Benders cuts. See Geoffrion and Graves (1974). The instances reported here are decomposition master problems.

• *TAM.* The annual decision on how much the Air Force should spend on aircraft and on munitions is of great interest to many people. How the Air Force staff develops information to support the decision has changed over the years. Currently, a linear program is being used by the Air Force Center for Studies and Analysis and is being tested by the Munitions Division of the Plans and Operations Directorate (AF/XOXFM) for munitions tradeoff analysis. The LP uses existing data and estimates of (1) aircraft and munition effectiveness, (2) target value, (3) attrition, (4) aircraft and munition costs, and (5) existing inventories of aircraft and munitions. Other factors considered are weather and length of the conflict. See Might (1987) and Jackson (1989).

• *PHOENIX.* A planning model for the multi-year, multi-billion dollar modernization of the U.S. Army's aging helicopter fleet. The mixed integer linear program employs a multiproduct production/inventory formulation with aged inventory. Goal constraints attempt to enforce fleet size, maximum age, and technology goals for each year and each of four aircraft missions, while also keeping expenditures within upper and lower limits. Additionally, combinatorial constraints and variables handle production line startup and shutdown costs, minimum and maximum production levels and requirements linking certain production lines. See Clemence, Teufert, Brown and Wood (1988) and Brown, Clemence, Teufert and Wood (1990).

• *EA6B.* Configures jammers of hostile radar on an EA-6B "Prowler" Naval electronic warfare aircraft. See Sterling (1990).

• *DEC.* Digital Equipment Corporation uses this model to determine worldwide manufacturing and distribution strategy for new products. This mixed integer, linear program suggests a production, distribution, and vendor network which minimizes cost and/or cumulative cycle times subject to constraints on estimated demand, local content, and joint

capacity, over multiple products, echelons, and time periods. Cost factors include fixed and variable production charges, distribution via multiple modes, taxes, duties and duty drawback, and inventory charges. See Harrison, Arntzen and Brown (1992).

• *AMMO 4H.* A four-commodity transshipment model for delivery over time of military products from production and storage locations to overseas locations to support theater operations is developed. The model covers five physical echelons, including production plants, storage depots, ports of embarkation, ports of debarkation and geographic field locations. Road, rail, sea and air transportation are modeled, and product demands are time-phased. Capacitation occurs primarily on sea and air links, and as throughput capacities on transfer points, requiring replication of some echelons. The objective of the model is to minimize deviation from on-time deliveries. See Staniec (1984).

• *BUSCH.* A model of brewery-to-wholesaler movements of beer for Anheuser-Busch. The model also includes some packaging decisions and is essentially a multicommodity flow model with joint capacity constraints arising from loading dock and inventory capacities as well as some managerial requirements. See Brown, Mamer, McBride and Wood (1992). The instance reported here is a small pilot model for the full-scale system with millions of variables which is solved directly, or by decomposition.

• *BAR.* A linear, mixed-integer multi-period production-inventory master planning model. See Harrison (1992).

Four implementations are compared: "XS" is an unadorned version of the X-System, an implementation of the Graves mutual primal-dual method with its GUB factorization disabled, while "XS(GUB)", "XS(PN)", and "XS(GN)" each employ the respective factorizations discussed here. To establish a frame of reference, performance of these implementations is compared with two well-known commercial solvers: IBM's Optimization Subroutine Library "OSL" (Release 2, 1991), and two versions of "CPLEX" (Version 1.2, 1990, and Version 2.0, 1992).

Ideally, one would develop four equivalent formulations of each model, each customized for its particular solver with the goal of inducing a large factored row set of the appropriate type. This approach is a consistent theme in the literature dealing with specialized algorithms and one that we strongly endorse. Alternate formulations of a model are often available, and it seems sensible to choose one that exploits as much as possible the strengths of the solver.

However, all of the models used here are "off-the-shelf" in the sense that they were developed at various times by various modelers, and alternate formulations are impracticable. Thus, the approach is to preserve a single, unfactored representation of each model, and attempt to identify favorable row structures through the use of heuristics. The procedure is based on the work of Brown and Thomen (1980), Brown and Wright (1983) and Brown, McBride and Wood (1985). The heuristics are greedy and myopic in the sense that they initially consider the entire row set of the problem, and discard one row at a time without backtracking until the remaining set satisfies the desired row factorization. This can be expected to confound or destroy structure introduced by the modeler. Although the auto-

matic factorization implementation has options to accept modeler guidance, the methods are compared here without this subjective complication. While these model-naive experiments yield interesting and useful observations about the implementations, they suffer for lack of guidance by a skilled modeler.

Table 9.1 shows the important structural information concerning the model instances to be solved.

Table 9.2 displays solution times for the sample problems. These CPU times exclude initial problem input, factored row identification, and final output — on average about 0.2 second per problem.

The original formulations of most of the test problems are strongly influenced by the modeler's solution strategy. For instance, TANKER is endowed with a GUB structure which places every binary variable in an associated set from which only one member can be chosen; this maximal GUB set is also sought for its tendency to yield nearly-integer

Table 9.1
Problem dimensions

	n	m	p_{GUB}/%	p_{PN}/%	p_{GN}/%	NZEL
GTE	6 624	960	909/95	909/95	922/96	58
INVEST	11 989	1 338	941/70	1 101/82	1 168/87	33
TANKER	7 598	83	32/39	32/39	66/80	31
HFDF	10 548	61	31/51	31/51	32/52	189
GAS PN A	27 884	6 848	4 345/63	5 934/87	5 976/87	37
GAS PN C	15 362	3 794	2 658/70	3 418/90	3 420/90	20
GAS PN E	5 102	1 184	434/37	877/74	883/75	7
GAS GN A	27 884	6 848	4 484/65	5 142/75	5 976/87	37
GAS GN C	15 362	3 794	2 664/70	3 084/81	3 420/90	20
KELLOGG 2	17 841	3 818	1 265/33	2 578/68	2 596/68	35
KELLOGG 3	27 490	5 727	2 295/40	3 867/68	3 892/68	54
KELLOGG 4	37 139	7 636	2 428/32	5 156/68	5 188/68	74
KELLOGG 5	46 788	9 545	3 388/36	6 445/68	6 484/68	93
ODS 1	11 568	3 023	528/17	540/18	558/18	21
ODS 3	23 993	594	490/82	490/82	490/82	68
TAM 5	10 531	438	102/23	132/30	162/37	94
TAM 8	6 104	420	118/28	154/37	196/47	49
TAM 12	17 793	629	177/28	231/37	294/47	165
PHOENIX 10	6 884	1 618	206/13	220/14	1 153/71	14
PHOENIX 30	17 212	4 305	293/07	303/07	3 604/84	48
EA6B	12 247	2 978	1 610/54	2 921/98	2 921/98	16
DEC	14 518	2 171	677/31	677/31	1 088/50	24
AMMO 4H	83 497	13 963	6 874/49	12 892/92	12 892/92	129
BUSCH 4	7 997	1 248	649/52	1 140/91	1 148/92	15
BAR	49 032	7 446	2 712/36	4 575/61	5 134/69	102

For each problem, the total number of structural variables is n, structural constraints m, GUB rows found by the identification heuristic p_{GUB}, pure network rows p_{PN}, generalized network rows p_{GN}, and thousands of nonzero technological coefficients NZEL. For example, GTE has 58 thousand nonzero technological coefficients for 6 624 variables; GTE can be viewed as having 960 explicit and no factored rows, or as 909/95% GUB-factored and $960 - 909 = 51$ explicit rows, as 909 PN-factored rows, or as 922 GN-factored and 38 explicit rows.

Table 9.2
Solution seconds

	AMDAHL 5995-700					486/33 MHz PC		
	X-System				IBM	XS	CPLEX	
	none	GUB	PN	GN	OSL	GN	1.2	2.0
GTE	9	8	8	8	43	41	65	58
INVEST	4	5	4	4	23	24	52	49
TANKER	8	10	10	8	6	48	8	9
HFDF	100	101	101	111	40	462	581	542
GAS PN A	2 376	778	50	57	291	321	1 714	1 221
GAS PN C	4	4	3	3	79	26	185	245
GAS PN E	72	33	4	6	13	33	165	50
GAS GN A	1 115	542	294	72	312	385	3 532	2 262
GAS GN C	4	4	3	3	71	26	186	236
KELLOGG 2	3	2	2	2	69	5	219	194
KELLOGG 3	5	5	5	5	144	9	513	440
KELLOGG 4	48	36	26	24	500	115	1 274	1 111
KELLOGG 5	1 122	1 459	320	248	1 210	926	2 615	2 243
ODS 1	6	3	2	5	125	22	1 042	414
ODS 3	6	6	6	6	23	38	58	56
TAM 5	55	60	45	40	44	231	151	86
TAM 8	24	14	14	17	21	69	77	71
TAM 12	108	112	88	106	101	312	416	378
PHOENIX 10	4	2	2	2	20	10	84	73
PHOENIX 30	41	25	26	9	335	69	1 171	1 239
EA6B	75	33	9	9	45	44	177	244
DEC	189	32	42	80	71	93	317	293
AMMO 4H	42	41	35	46	1 000	277	1 762	1 597
BUSCH 4	5	5	4	4	26	34	55	46
BAR	290	212	106	114	382	480	1 366	1 222

An AMDAHL 5995-700 running under IBM VM/CMS/XA with IBM VS FORTRAN 2.3.0 is used to render performance in CPU-seconds accurate to the precision shown for the basic "XS"-system using no factorization, compared with dynamic factorizations of "XS(GUB)", pure network "XS(PN)", and generalized network "XS(GN)" rows. "IBM-OSL" shows primal simplex performance on the same computer of the IBM Optimization Subroutine Library, Release 2 (1991). "486/33" shows the clock-time performance of "XS(GN)" on a microcomputer (33 MHz Intel 486 with 32 MB RAM) followed by that of "CPLEX" (Version 1.2) (1990) and of "CPLEX" (Version 2.0) (1992) on the same microcomputer.

solutions to the linear program. AMMO 4H is a multicommodity capacitated transshipment problem and thus is best suited to a pure network factorization; it was originally solved by dual decomposition rendering pure network sub-problems. PHOENIX is a multicommodity equipment replacement model closely following the generalized network factorization paradigm.

The row structures in Table 9.2 have been found without modeler help by our automatic identification heuristic, yet they corroborate the modelers' intentions. TANKER reveals the same pure network rows as the GUB rows, or a generalized network constructed by iden-

tifying one additional row to be paired with each GUB row. PHOENIX exhibits a dominant structure that is clearly a generalized network.

One would anticipate the factorization exploiting the dominant row structure to win computation tests. This is wrong more often than right. Table 9.2 shows that the more general factorization dominates the less general, with few exceptions: GTE's relatively large GUB set, and the large pure networks in GAS PN A, GAS PN E, and AMMO 4H seem to satisfy our prior bias toward model-dictated factorizations.

Table 9.2 also suggests that our myopic use of heuristics to automatically identify factored structure has its pitfalls. In a number of problem instances, we identify significantly larger factored sets with the more general factorizations, yet we enjoy little improvement in computation times (e.g., INVEST, ODS 3, TAM 8). This suggests that the "quality" of a row factorization is not completely specified by its size.

We have pursued this notion by inviting some of the modelers to guide our identification heuristic to precisely the row sets they intended. Some of the results have been striking: Wood (1989) reports significant improvements for problems in the GAS system.

Table 9.3
Maximum number of elements in explicit transformation kernel

	XS	XS(GUB)	XS(PN)	XS(GN)
GTE	13	0	0	0
INVEST	9	1	1	1
TANKER	1	1	1	0
HFDF	3	1	1	1
GAS PN A	1 550	891	30	54
GAS PN C	18	5	0	0
GAS PN E	263	110	7	5
GAS GN A	1 470	758	340	59
GAS GN C	19	7	1	0
KELLOGG 2	6	2	0	0
KELLOGG 3	9	4	0	0
KELLOGG 4	79	41	10	11
KELLOGG 5	1 299	1 015	138	128
ODS 1	9	1	1	5
ODS 3	0	0	0	0
TAM 5	28	22	15	18
TAM 8	20	15	8	10
TAM 12	38	42	16	21
PHOENIX 10	32	21	21	1
PHOENIX 30	169	157	185	1
EA6B	1 155	270	0	0
DEC	218	39	49	46
AMMO 4H	235	92	0	0
BUSCH 4	10	5	0	0
BAR	290	163	70	41

The number of nonzero elements (in nearest thousands) in the explicit transformation kernel $\tilde{A}_{11}^{-1}$ gives some indication of how much information is not captured by factorization, as well as an idea of relative storage requirements.

Table 9.4
Binding explicit constraints at optimality

	Binding/%	GUB/%	PN/%	GN/%
GTE	554/58	20/02	20/02	18/02
INVEST	763/57	201/15	195/15	162/12
TANKER	51/61	60/72	39/47	9/11
HFDF	58/95	29/48	29/48	28/46
GAS PN A	3 068/45	1 953/29	299/04	349/05
GAS PN C	2 324/61	850/22	68/02	90/02
GAS PN E	901/76	573/48	90/08	79/07
GAS GN A	3 064/45	1 854/27	1 155/17	359/05
GAS GN C	2 323/61	861/23	402/11	89/02
KELLOGG 2	1 950/51	1 015/27	92/02	118/03
KELLOGG 3	2 942/51	1 368/24	148/03	183/03
KELLOGG 4	4 136/54	2 491/33	391/05	452/06
KELLOGG 5	5 270/55	2 305/24	592/06	617/06
ODS 1	361/12	83/03	76/03	117/04
ODS 3	410/69	0/00	0/00	0/00
TAM 5	264/60	227/52	168/38	186/42
TAM 8	279/66	240/57	142/34	175/42
TAM 12	412/66	352/53	205/33	251/40
PHOENIX 10	1 098/68	1 096/68	1 090/67	80/05
PHOENIX 30	3 298/77	3 236/75	3 239/75	110/03
EA6B	2 939/99	1 334/45	26/01	27/01
DEC	1 328/61	736/34	726/33	421/19
AMMO 4H	6 686/46	3 153/22	13/00	14/00
BUSCH 4	840/67	481/39	5/00	8/01
BAR	4 687/63	3 250/44	1 895/25	1 450/19

Not all constraints are binding at optimality. The first column lists the number of binding explicit constraints and expresses this as a percentage of all constraints; the following columns display the number of binding explicit constraints and their corresponding percentages under the alternate factorizations.

A few of our corresponding modelers have had the opportunity to build models from scratch with a particular factorization in mind. This admits model coercion and a wide range of well-known reformulation methods which we think can materially change both the size and quality of the result. Their early reports show promise. Among the models discussed here, the GAS and KELLOGG systems have been subsequently reformulated to enhance generalized networks, GTE has been re-engineered to further accentuate its dominant GUB set, and TANKER-like and many ODS models have been moved to a micro-computer; all these models are now larger, but much easier to solve.

It is surprising and encouraging that the transition to more general factorizations seldom degrades performance much, even when few additional factored rows are won by the increased generality. This contradicts popular folklore that the more general factorizations demand substantial, if not overwhelming increases in the resulting sizes of the factored structures. In fact, computational testing reported by others has usually been limited to models in which the number of explicit rows is in the range of one to twenty (e.g., Chen and Saigal, 1977; Glover, Karney, Klingman and Russell, 1978; Glover and Klingman,

1981). Our results are all the more remarkable given the lack of guidance from the modeler for the "intended" row factorization.

Table 9.3 shows the maximum size of the $\tilde{A}_{11}^{-1}$ in terms of its nonzero elements. We see that the maximum size of the explicit transformation kernel tends to decrease as the generality of the factorization increases. Recalling the definition of the explicit transformation kernel,

$$\tilde{A}_{11}^{-1} = (E_{11} - E_{12}F_{22}^{-1}F_{21})^{-1},$$

this trend is as we would expect. Each potentially binding explicit row which can be converted to a factored row reduces the likely size of $\tilde{A}_{11}^{-1}$. Also, the density of the term $-E_{12}F_{22}^{-1}F_{21}$ generally increases with the density of F_{22}^{-1}. For F_{22} k-by-k, the number of nonzeros in F_{22}^{-1} for the GUB factorization is k, for pure networks perhaps as large $\frac{1}{2}k^2$, and for generalized networks as large as k^2. There are some exceptions to this trend in Table 9.3, especially in model instances in which the size of the (GN) factored row set is not significantly larger than that of (PN). This is because for a given factored kernel F_{22}, the exact (PN) representation of F_{22}^{-1} is generally more sparse than that of the less exact floating-point representation of (GN).

It is usually the case that many constraints are not binding at optimality, as can be seen in Table 9.4. A distinguishing feature of dynamic factorization is the ability to limit attention to binding constraints, handling binding factored constraints with great efficiency, and working with a relatively small number of binding explicit constraints. Explicit binding constraints on the order of a few thousand, or less, are quite manageable. This is well beyond the size of previously reported implementations.

10. Conclusions

Previous research by others generally suggests that specialized algorithms such as those presented here are useful only when the factored structure completely dominates. There are even reports of algorithms for solving problems having a single unfactored (explicit) constraint (Hultz and Klingman, 1978; Klingman and Russell, 1978). When implementations have been reported, problem suites have been limited to instances having a very small number of explicit constraints, typically in the range from one to twenty (Chen and Saigal, 1977; Glover, Karney, Klingman and Russell, 1978; Glover and Klingman, 1981). The consensus seems to be that such algorithms are quite delicate, and deserve to be viewed as specialized algorithms, useful only for solving very special problem instances.

We refute this view. Dynamic factorization is competitive with commercial-quality optimization systems on every model instance we have tested.

The development here stresses the similarities among the algorithms and the natural extensions leading from one to the next. This is in contrast to the development reported for similar, non-dynamic algorithms (e.g., Dantzig and Van Slyke, 1967; Klingman and Rus-

sell, 1978; and Hultz and Klingman, 1978) in which the specifics of the individual algorithm obscure the generality of the approach. The conceptual difference between our algorithms is seen to be largely isolated to the structure of a single algebraic entity, the factored kernel. By abstracting the structure of the factored kernel and concentrating on the general algorithm design, the versatility and flexibility of this approach is clarified.

The algorithmic development leads directly to an implementation. The resulting software suite exhibits a "single system image". The modularity of the algorithm allows the definition of an "abstract data type" (see, e.g., Aho, Hopcroft and Ullman, 1974) which isolates the data structures and update procedures for the factored kernel from the rest of the implementation. Each factorization is seamlessly integrated within the system design.

The early 1980s produced a great deal of research on automatic identification of special structure in LP models (see, e.g., Gunawardane and Schrage, 1977; Glover, 1980; Schrage, 1981; Brown, McBride and Wood, 1985; and Bixby and Fourer, 1986). We have incorporated the most useful of these ideas into our implementation, and we have what we believe to be the first complete implementation which supports automatic identification of factored row sets. This capability may be used to identify new factored structure or to validate or augment a modeler-provided recommendation. When faced with the choice of either solving an unfactored model instance or automatically identifying a factored structure and then using the corresponding solver, our results show that the latter is nearly always to be preferred. Modelers have conducted extensive additional computational experimentation with the X-System not reported in this paper. These results suggest that in addition to the quantity of factored rows, the quality of these rows influences the performance of factorization algorithms. While not well understood, it is clear that the myopic approach of our heuristics is no substitute for the modeler's guidance in identifying factored structure.

Processing networks (Koene, 1982) are network models which allow proportional flow restrictions on the arcs entering or leaving some nodes. One formulation of such a model results in a pure or generalized network structure with a set of complicating columns. Chen and Engquist (1986) propose a primal partitioning algorithm for solving processing network problems. An alternate formulation yields a pure or generalized network structure with complicating rows: this is precisely the structure dealt with here.

The multicommodity capacitated transshipment problem (MCTP) has been the subject of much research over the years, and a number of specialized algorithms have been proposed to solve it (see, e.g., Assad, 1978, or Kennington, 1978). The usual MCTP formulation is a pure network which each commodity uses independently in its own flow model, but with side constraints on the total common flow of all commodities over some of the network arcs. The side constraints form a GUB row set, while the rest of MCTP forms a pure network; either view might be preferred depending upon size of the common network, the number of side constraints, and the number of commodities. In our experience, the network factorization usually dominates the GUB factorization, and the pure network factorization presented here is a powerful technique for solving MCTPs. As an experiment, we customized our (PN) implementation for MCTP to exploit the special structure of the explicit side

constraints. This highly-specialized implementation performed no better on AMMO 4H, and we now believe that this would be true for most MCTPs.

There are problems which would exhibit a large factored row structure if not for a set of complicating columns (e.g., see Brown, McBride and Wood, 1985). One would expect the structure of the factored kernel to be dominated by that of the predominant row structure, with only occasional complications due to the exceptional columns. One might allow for this exceptional structure in the factored kernel by identifying it "on-the-fly" as the algorithm progresses, and preserving the sanctity of the core factorization. Though conceptually simple, some iterations of this algorithm would border on the spectacular. This approach may be thought of as a hybrid between the dynamic factorization developed here and dynamic basis triangulation methods (see, e.g., Hellerman and Rarick, 1971, 1972; Saunders, 1976, and McBride, 1980).

Dynamic extrinsic factorization is subsumed by the algorithms presented in this paper if we activate functions in the update analogous to the secondary exchanges now employed. Essentially all that has to be done is ensure that successive factored components retain their stipulated special structure. We speculate that this will work best in cases where model structure is amenable, and quite likely will require some model-specific customization to perform well on difficult models. We have limited our experimentation to those static extrinsic cases which we believe to be most generally useful.

References

A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Menlo Park, CA, 1974).

A. Assad, "Multicommodity network flows – A survey," *Networks* 8 (1978) 37–91.

W. Avery, G.G. Brown, J.A. Rosenkranz and R.K. Wood, "Optimization of purchase, storage and transmission contracts for natural gas utilities," *Operations Research*, 40(3) (1992) 446–462.

M.L. Balinski and R.L. Gomory, "A mutual primal–dual simplex method," in: *Recent Advances in Mathematical Programming* (McGraw-Hill, New York, 1963).

J.F. Benders, "Partitioning procedure for solving mixed-variables programming problems," *Numerische Mathematik* 4 (1962) 238–252.

R.E. Bixby and R. Fourer, "Finding embedded network rows in linear programs I: Extraction heuristics," Report No. 86437-OR, Oekonometrie und Operations Research, Bonn University (Bonn, 1986).

G.H. Bradley, "Optimization of capital portfolios," *Proceedings of the National Communications Forum 86* (1986) pp. 11–17.

G.H. Bradley, G.G. Brown and G.W. Graves, "Design and implementation of large-scale primal transshipment algorithms," *Management Science* 24(1) (1977) 1–34.

A.L. Brearley, G. Mitra and H.P. Williams, "Analysis of mathematical programming problems prior to applying the simplex algorithm," *Mathematical Programming* 8 (1978) 54–83.

G.G. Brown, R.D. Clemence Jr., W.R. Teufert and R.K. Wood, "An optimization model for modernizing the army's helicopter fleet," *Interfaces* 21(4) (1990) 39–52.

G.G. Brown, D.A. Drake, A.B. Marsh and A. Washburn, "Mathematical methods applied to managing a system of direction-finding receivers," Military Operations Research Society (Annapolis, MD, 1990).

G.G. Brown and G.W. Graves, "Elastic programming: A new approach to large-scale mixed-integer optimization," presented at ORSA/TIMS meeting, Las Vegas, NV (1975).

G.G. Brown, G.W. Graves and D. Ronen, "Scheduling ocean transportation of crude oil," *Management Science* 33(3) (1987) 335–346.

G.G. Brown, J.W. Mamer, R.D. McBride and R.K. Wood, "Solving a large-scale generalized multi-commodity flow problem," ORSA/TIMS (San Francisco, CA, 1992).

G.G. Brown and R.D. McBride, "Solving generalized networks," *Management Science* 30(12) (1984) 1497–1523.

G.G. Brown, R.D. McBride and R.K. Wood, "Extracting embedded generalized networks from linear programming problems," *Mathematical Programming Study* 32 (1985) 11–31.

G.G. Brown and D. Thomen, "Automatic identification of generalized upper bounds in large-scale optimization models," *Management Science* 26(11) (1980) 1166–1184.

G.G. Brown and W. Wright, "Automatic identification of embedded network rows in large-scale optimization models," *Mathematical Programming* 29 (1984) 41–56.

A. Charnes and C.E. Lemke, "Computational theory of linear programming, I: The bounded variables problem," ONR Research Memorandum 10, Graduate School of Industrial Administration, Carnegie Institute of Technology (Pittsburgh, PA, 1952).

C. Chen and M. Engquist, "A primal simplex approach to pure processing networks," *Management Science* 32(12) (1986) 1582–1598.

S. Chen and R. Saigal, "A primal algorithm for solving a capacitated network flow problem with additional linear constraints," *Networks* 7 (1977) 59–79.

R.D. Clemence Jr., W.R. Teufert, G.G. Brown and R.K. Wood, "Phoenix: Developing and evaluating army aviation modernization policies using mixed integer linear programming," *27th U.S. Army Operations Research Symposium* (Fort Lee, VA, 1988).

CPLEX Optimization, Inc., *Using the CPLEX(TM) Linear Optimizer (Version 1.2)* (Incline Village, NV, 1990).

CPLEX Optimization, Inc., *Using the CPLEX(TM) Linear Optimizer and CPLEX(TM) Mixed Integer Optimizer (Version 2.0)* (Incline Village, NV, 1992).

G.B. Dantzig, "Notes on linear programming: Parts VIII, IX, X-upper bounds, secondary constraints, and block triangularity in linear programming," Research Memorandum RM-1367, The Rand Corporation (Santa Monica, CA, 1954).

G.B. Dantzig, *Linear Programming and Extensions* (Princeton University Press, Princeton, NJ, 1963).

G.B. Dantzig and R.M. Van Slyke, "Generalized upper bounding techniques," *Journal of Computer and System Sciences* 1 (1967) 213–226.

G.B. Dantzig and P. Wolfe, "Decomposition principal for linear programming," *Operations Research* 8(1) (1960) 101–111.

A.M. Geoffrion and G.W. Graves, "Multicommodity distribution system design by Benders decomposition," *Management Science* 29(5) (1974) 822–844.

F. Glover, "Transformations enlarging the network portion of a class of LP/embedded generalized networks," MSRS 80-1, University of Colorado (Boulder, CO, 1980).

F. Glover, J. Hultz, D. Klingman and J. Stutz, "Generalized networks" A fundamental computer-based planning tool," *Management Science* 24(12) (1978) 1209–1220.

F. Glover, D. Karney, D. Klingman and R. Russell, "Solving singly constrained transshipment problems," *Transportation Science* 12(4) (1978) 277–297.

F. Glover and D. Klingman, "The simplex SON algorithm for LP/embedded network problems," *Mathematical Programming Study* 15 (1981) 148–176.

G.H. Golub and C.F. Van Loan, *Matrix Computations* (The Johns Hopkins University Press, Baltimore, MD, 1983).

G.W. Graves, "A complete constructive algorithm for the general mixed linear programming problem," *Naval Research Logistics Quarterly* 12(1) (1965) 1–14.

G.W. Graves and R.D. McBride, "The factorization approach to large-scale linear programming," *Mathematical Programming* 10 (1976) 91–110.

H.J. Greenberg and D.C. Rarick, "Determining GUB sets via an invert agenda algorithm," *Mathematical Programming* 7 (1974) 240–244.

G. Gunawardane and L. Schrage, "Identification of special structure constraints in linear programs," University of Chicago (Chicago, IL, 1977).

T.P. Harrison, G.H. Bradley and G.G. Brown, "Capital allocation and project selection via decomposition," presented at CORS/TIMS/ORSA meeting, Vancouver, BC (1989).

T.P. Harrison, B.C. Arntzen and G.G. Brown, "Global manufacturing strategy analysis," presented at ORSA/TIMS meeting, Orlando, FL (1992).

T.P. Harrison, Private communication (1992).

J.K. Hartman and L.S. Lasdon, "A generalized upper bounding method for doubly coupled linear programs," Technical Memorandum No. 140 (1970).

J.K. Hartman and L.S. Lasdon, "A generalized upper bounding algorithm for multicommodity network flow problems," *Networks* 1 (1972) 333–354.

R.V. Helgason and J.L. Kennington, "A product form representation of the inverse of a multicommodity cycle matrix," *Networks* 7 (1977) 297–322.

E. Hellerman and D. Rarick, "Reinversion and the preassigned pivot procedure," *Mathematical Programming* 1 (1971) 195–216.

E. Hellerman and D. Rarick, "The partitioned preassigned pivot procedure (P^4)," in: D.J. Rose and P.A. Willoughby, eds., *Sparse Matrices and their Applications* (Plenum, New York, 1972) pp. 67–76.

J. Hultz and D. Klingman, "Solving constrained generalized network problems," Research Report CCS 257, Center for Cybernetic Studies, University of Texas at Austin (Austin, TX, 1976).

J. Hultz and D. Klingman, "Solving singularly constrained generalized network problems," *Applied Mathematics and Optimization* 4 (1978) 103–119.

IBM Corporation, *Optimization Subroutine Library Guide and Reference Release 2* (Kingston, NY, 1991).

J.A. Jackson, "A taxonomy of advanced linear programming techniques and the theater attack model," Master's Thesis, Air Force Institute of Technology, Air University (Wright–Patterson Air Force Base, OH, 1989).

J.L. Kennington, "Solving multicommodity transportation problems using a primal partitioning simplex technique," *Naval Research Logistics Quarterly* 24(2) (1977) 309–325.

J.L. Kennington, "A survey of linear cost multicommodity network flows," *Operations Research* 26 (1978) 209–236.

D. Klingman and R. Russell, "On solving constrained transportation problems," *Operations Research* 23(1) (1975) 91–107.

D. Klingman and R. Russell, "A streamlined simplex approach to the singly constrained transportation problem," *Naval Research Logistics Quarterly* 25(4) (1978) 681–696.

J. Koene, "Minimal cost flow in processing networks, a primal approach," Ph.D. Thesis, Eindhoven University of Technology (Eindhoven, The Netherlands, 1982).

R.D. McBride, "Factorization in large-scale linear programming," Working Paper No. 22, University of California (Los Angeles, CA, 1972).

R.D. McBride, "A bump triangular dynamic factorization algorithm for the simplex method," *Mathematical Programming* 18 (1980) 49–61.

R.D. McBride, "Solving embedded generalized network problems," *European Journal of Operational Research* 21 (1985) 82–92.

R.D. McBride, Private communication (1989).

R.J. Might, "Decision support for aircraft and munitions procurement," *Interfaces* 17(5) (1987) 55–63.

B.A. Murtagh and M.A. Saunders, "MINOS user's guide," Technical Report SOL 77-9, Systems Optimization Laboratory, Department of Operations Research, Stanford University (Stanford, CA, 1977).

M.P. Olson, "Dynamic factorization in large-scale optimization," Doctoral Dissertation, Naval Postgraduate School (Monterey, CA, 1989).

S. Powell, "A development of the product form algorithm for the simplex method using reduced transformation vectors," *Mathematical Programming* 9 (1975) 93–107.

J.B. Rosen, "Primal partition programming for block diagonal matrices," *Numerical Mathematics* 6 (1964) 250–260.

M.A. Saunders, "A fast, stable implementation of the simplex method using Bartels–Golub Updating," in: J.R. Bunch and D.J. Rose, eds., *Sparse Matrix Computations* (Academic Press, New York, 1976) pp. 213–226.

L. Schrage, "Implicit representation of variable upper bounds in linear programming," *Mathematical Programming* 4 (1975) 118–132.

L. Schrage, "Implicit representation of generalized variable upper bounds in linear programming," *Mathematical Programming* 14 (1978) 11–20.

L. Schrage, "Some comments on hidden structure in linear programs," in: H.J. Greenberg and I. Maybee, eds., *Computer-assisted Analysis and Model Simplification* (Academic Press, New York, 1981) pp. 389–395.

C.J. Staniec, "Design and solution of an ammunition distribution model by a resource-directive multicommodity network flow algorithm," Master's Thesis, Naval Postgraduate School (Monterey, CA, 1984).

J. Sterling, "An EA-6B transmitter loading and assignment model," Master's Thesis, Naval Postgraduate School (Monterey, CA, 1990).

M.J. Todd, "Large-scale linear programming: Geometry, working bases and factorization," *Mathematical Programming* 26(1) (1983) 1–20.

R.K. Wood, Private communication (1989).

G. Zoutendijk, "A product-form algorithm using contracted transformation vectors," in: J. Abadie, ed., *Integer and Nonlinear Programming* (North-Holland, Amsterdam, 1970).